Final Year B. Tech., Sem VI 2021-22

High Performance Computing Lab

PRN No: 2019BTECS00069

Name: Vaishnavi Disale

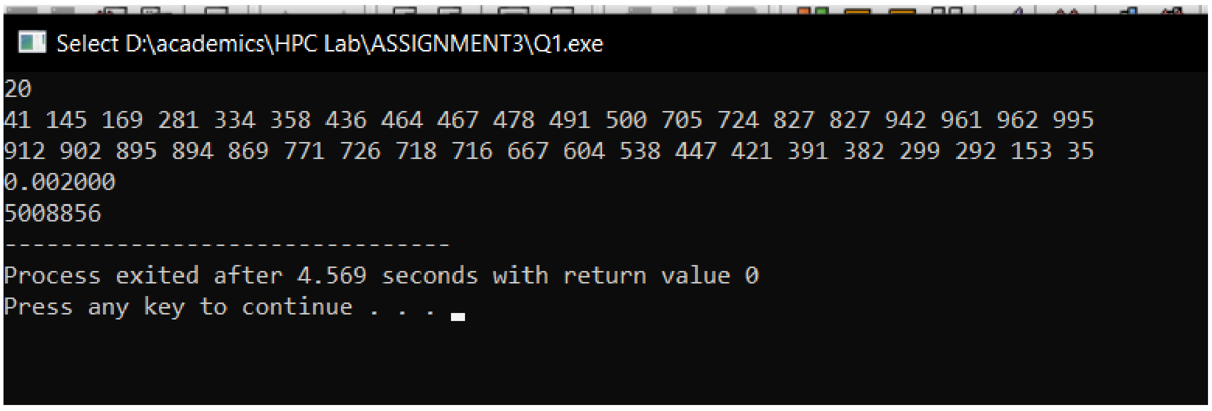
Batch: B3

Assignment: 3

Q1: Analyse and implement a Parallel code for below program using OpenMP.

```
// C Program to find the minimum scalar product of two vectors (dot product)
#include<bits/stdc++.h>
#include <omp.h>

using namespace std;

int sort(int arr[], int n)
{
    int i, j;
    #pragma omp parallel shared(arr) private(j)
    #pragma omp for schedule(dynamic)
    for (i = 0; i < n-1; i++)
        for (j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
            {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
}

int sort_des(int arr[], int n)
{
    int i,j;
    #pragma omp parallel shared(arr) private(j)
    #pragma omp for schedule(dynamic)
    for (i = 0; i < n; ++i)
    {
        for (j = i + 1; j < n; ++j)
        {
            if (arr[i] < arr[j])
            {
                int a = arr[i];
                arr[i] = arr[j];
                arr[j] = a;
            }
        }
    }
}

int main()
{
    //fill the code;
    int i,tid,n,psum;
    int threads = 4;
    cout<<"Enter Size of Array: ";
    cin>>n;
    int arr1[n], arr2[n];
    cout<<"Enter Elements of First Array:\n";
    for(i = 0; i < n ; i++)
    {
        cin>>arr1[i];
    }
    cout<<"Enter Elements of Second Array:\n";
    for(i = 0; i < n ; i++)
    {
        cin>>arr2[i];
    }
    sort(arr1, n);
    sort_des(arr2, n);
    int sum = 0;
    #pragma omp parallel private(i,tid,psum) num_threads(threads)
    {
        psum=0;
        tid = omp_get_thread_num();
        #pragma omp for reduction(+:sum)
        for(int i=0; i<n; i++)
        {
            sum += arr1[i] * arr2[i];
            psum+=sum;
        }
        printf("Thread %d partial sum = %d\n",tid,psum);
    }
    cout<<"Sum: "<<sum<<endl;

    return 0;
}
```

Output:



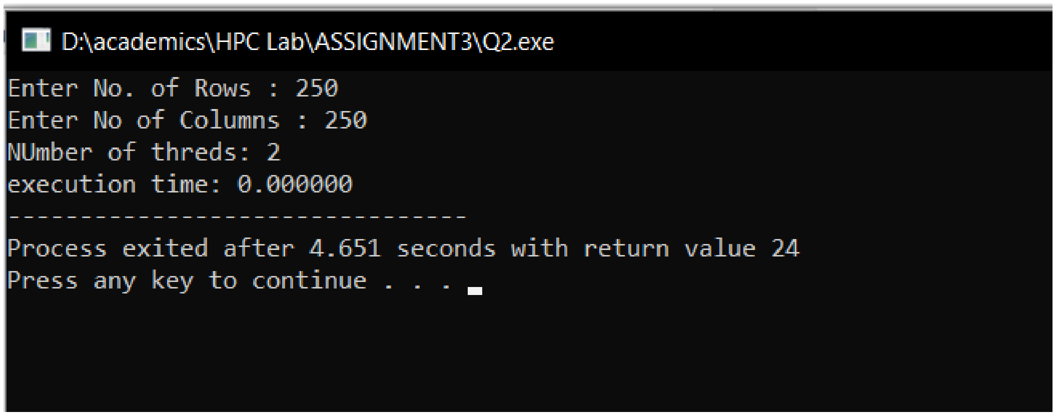
Select D:\academics\HPC Lab\ASSIGNMENT3\Q1.exe

```
20
41 145 169 281 334 358 436 464 467 478 491 500 705 724 827 827 942 961 962 995
912 902 895 894 869 771 726 718 716 667 604 538 447 421 391 382 299 292 153 35
0.002000
5008856
--------------------------------
Process exited after 4.569 seconds with return value 0
Press any key to continue . . .
```

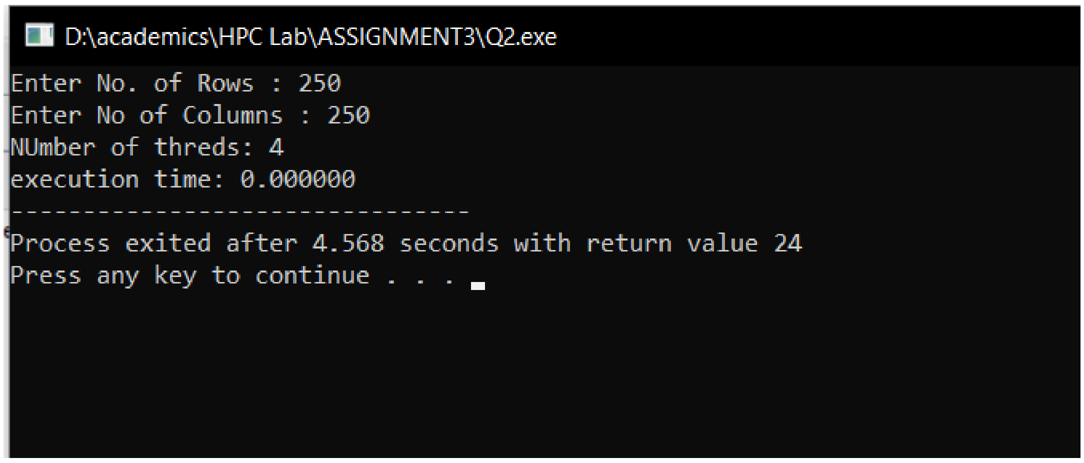
Q2: Write OpenMP code for two 2D Matrix addition, vary the size of your matrices from 250, 500, 750, 1000, and 2000 and measure the runtime with one thread (Use functions in C in calculate the execution time or use GPROF)

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main()
{

    int row, col;

    printf("Enter No. of Rows : ");
    scanf("%d", &row);

    printf("Enter No of Columns : ");
    scanf("%d", &col);

    int a[row][col], b[row][col], c[row][col];
    int i, j;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            a[i][j] = rand() * 1000;
            b[i][j] = rand() * 1000;
        }
    }

    omp_set_num_threads(8);

    double startTime = omp_get_wtime();

// int i,j;
#pragma omp parallel for shared(a, b, c, row, col) schedule(static, row / 8)
collapse(2)
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {

            c[i][j] = a[i][j] + b[i][j];
        }
    }
    double endTime = omp_get_wtime();

    printf("execution time: %f", endTime - startTime);
}
```

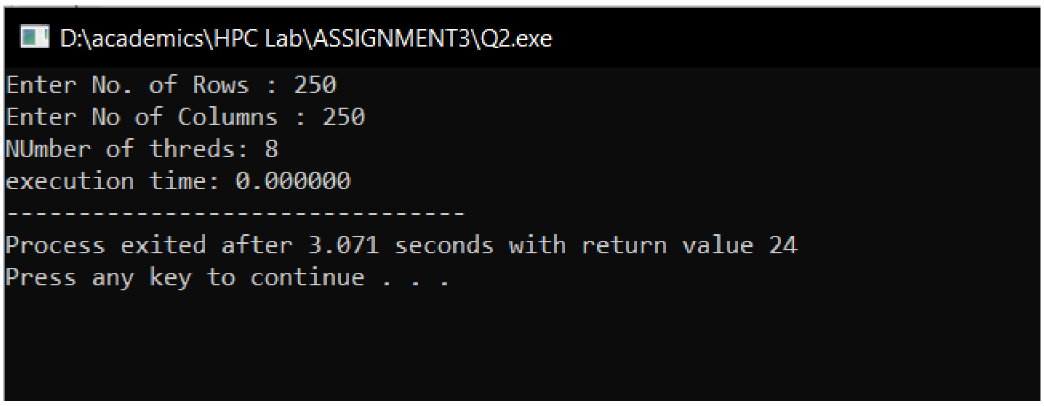
i. For each matrix size, change the number of threads from 2,4,8., and plot the speedup versus the number of threads.



```
D:\academics\HPC Lab\ASSIGNMENT3\Q2.exe
Enter No. of Rows : 250
Enter No of Columns : 250
NUmber of threds: 2
execution time: 0.000000
--------------------------------
Process exited after 4.651 seconds with return value 24
Press any key to continue . . .
```



```
D:\academics\HPC Lab\ASSIGNMENT3\Q2.exe
Enter No. of Rows : 250
Enter No of Columns : 250
NUmber of threds: 4
execution time: 0.000000
--------------------------------
Process exited after 4.568 seconds with return value 24
Press any key to continue . . .
```

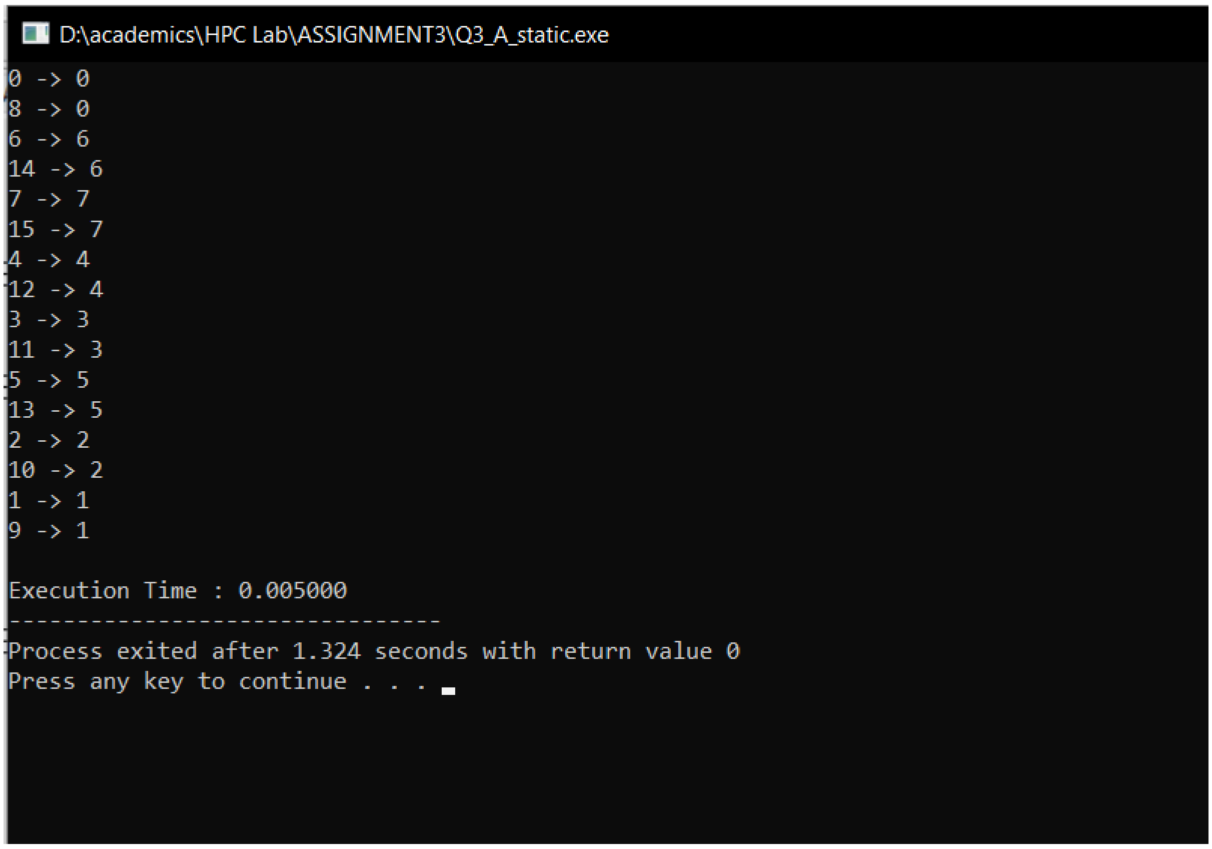


```
D:\academics\HPC Lab\ASSIGNMENT3\Q2.exe
Enter No. of Rows : 250
Enter No of Columns : 250
NUmber of threds: 8
execution time: 0.000000
--------------------------------
Process exited after 3.071 seconds with return value 24
Press any key to continue . . .
```

Q3. For 1D Vector (size=200) and scalar addition, Write a OpenMP code with the following:

i. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup.

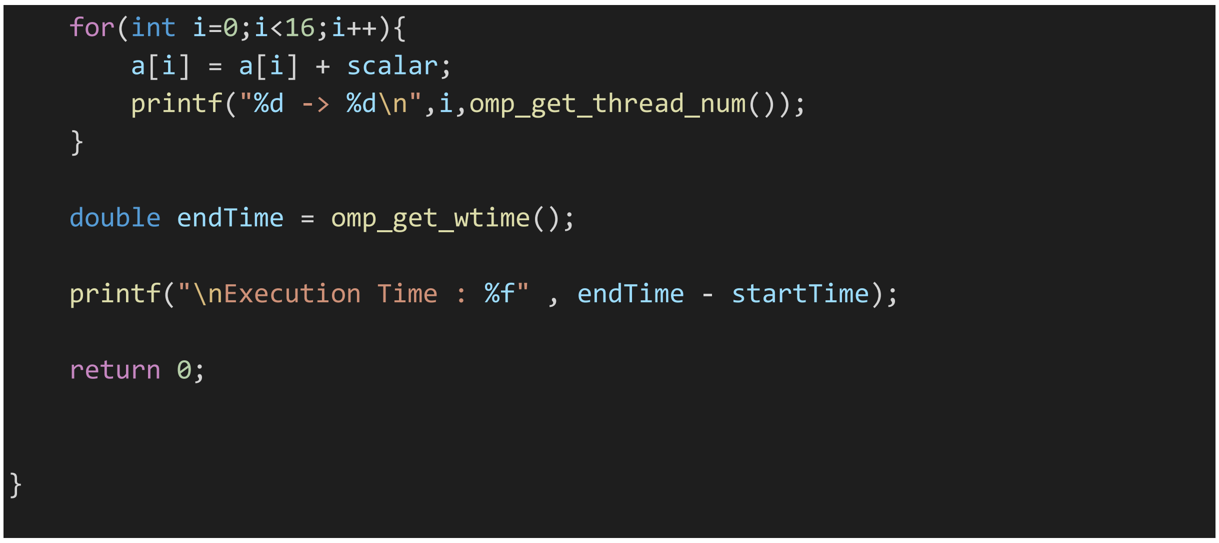
```
#include<omp.h>
#include<stdio.h>
#include<stdlib.h>


int main(){

    int a[200];

    for(int i=0;i<200;i++){
        a[i] = rand()*1000;
    }

    int scalar  = 10;


    double startTime = omp_get_wtime();

    #pragma omp parallel for schedule(static , 1)
    for(int i=0;i<16;i++){
        a[i] = a[i] + scalar;
        printf("%d -> %d\n",i,omp_get_thread_num());
    }

    double endTime = omp_get_wtime();

    printf("\nExecution Time : %f" , endTime - startTime);

    return 0;


}
```


D:\academics\HPC Lab\ASSIGNMENT3\Q3_A_static.exe

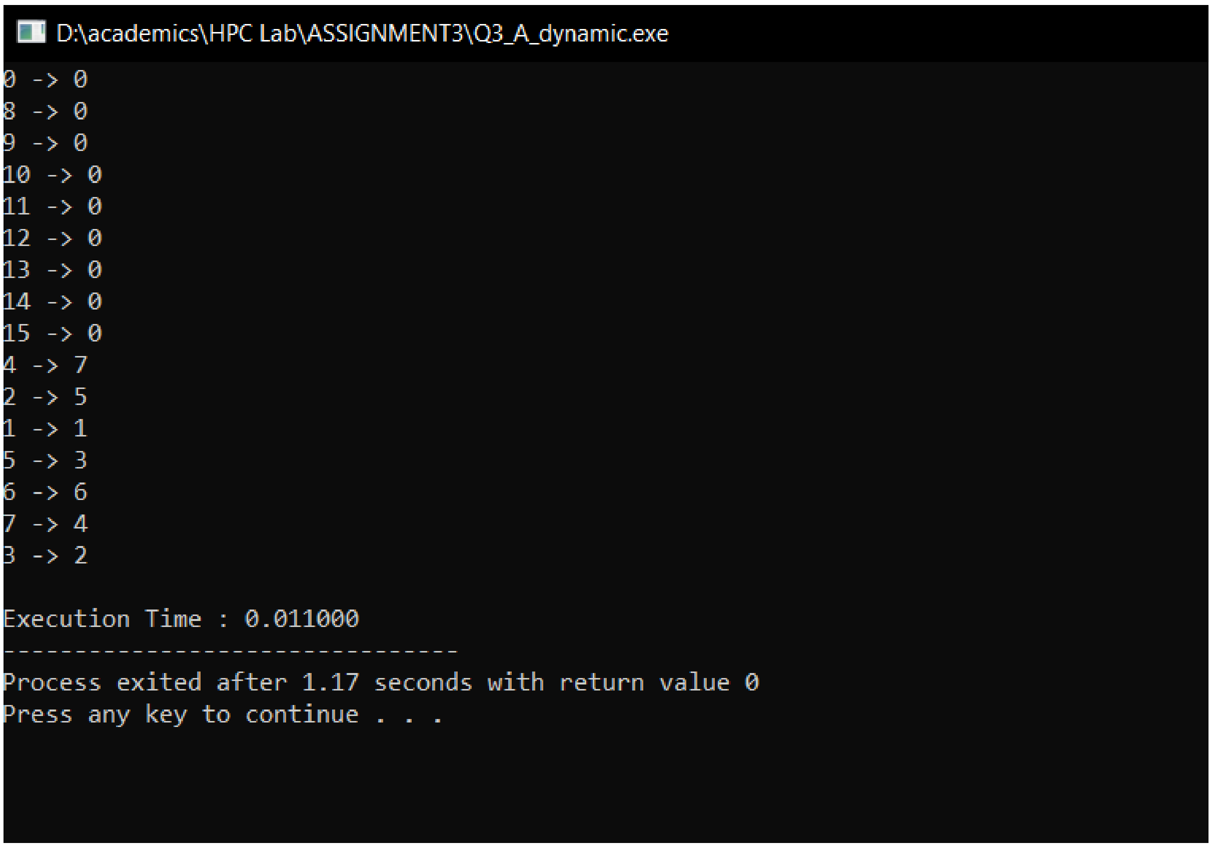
```
0 -> 0
8 -> 0
6 -> 6
14 -> 6
7 -> 7
15 -> 7
4 -> 4
12 -> 4
3 -> 3
11 -> 3
5 -> 5
13 -> 5
2 -> 2
10 -> 2
1 -> 1
9 -> 1

Execution Time : 0.005000
--------------------------------
Process exited after 1.324 seconds with return value 0
Press any key to continue . . .
```

ii. Use DYNAMIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup.

```c
#include<omp.h>
#include<stdio.h>
#include<stdlib.h>


int main(){

    int a[200];

    for(int i=0;i<200;i++){
        a[i] = rand()*1000;
    }

    int scalar  = 100;


    double startTime = omp_get_wtime();

    #pragma omp parallel for schedule(dynamic , 1)
```



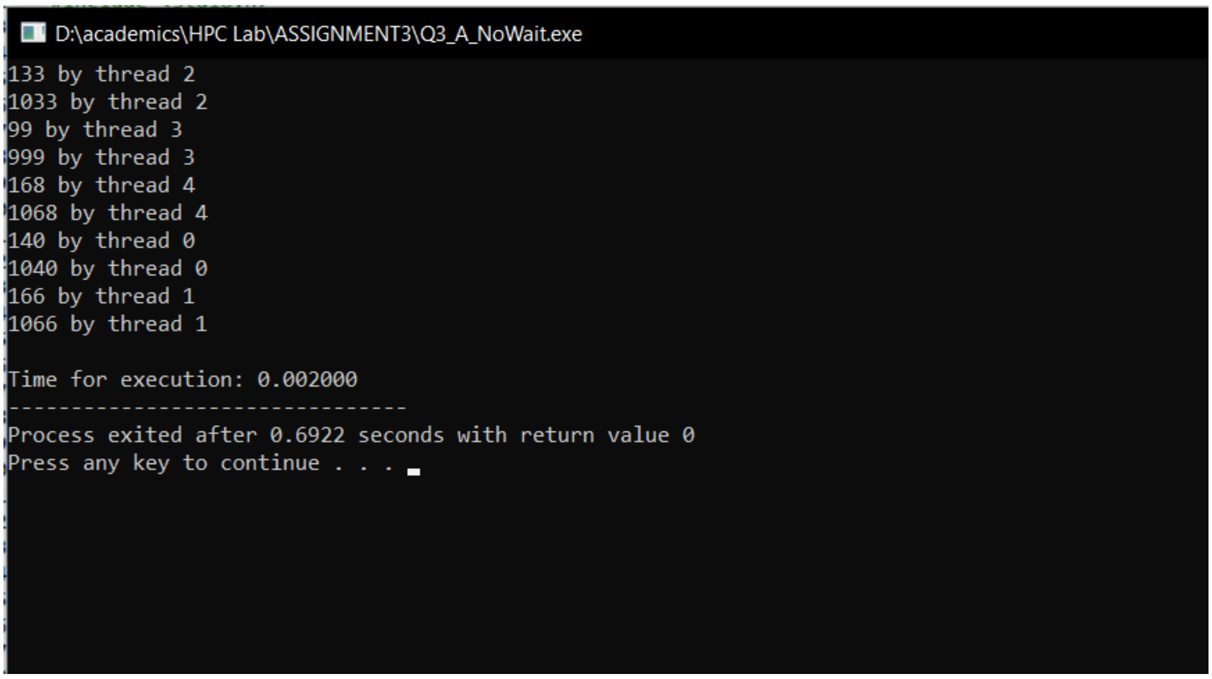
```
    for(int i=0;i<16;i++){
        a[i] = a[i] + scalar;
        printf("%d -> %d\n",i,omp_get_thread_num());
    }

    double endTime = omp_get_wtime();

    printf("\nExecution Time : %f" , endTime - startTime);

    return 0;


}
```



```
D:\academics\HPC Lab\ASSIGNMENT3\Q3_A_dynamic.exe
0 -> 0
8 -> 0
9 -> 0
10 -> 0
11 -> 0
12 -> 0
13 -> 0
14 -> 0
15 -> 0
4 -> 7
2 -> 5
1 -> 1
5 -> 3
6 -> 6
7 -> 4
3 -> 2

Execution Time : 0.011000
--------------------------------
Process exited after 1.17 seconds with return value 0
Press any key to continue . . .
```

iii. Demonstrate the use of nowait clause.

```c
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
    int n = 5, i ,j=99;
    int arr1[n], answer[n],answer2[n];
    for(i = 0; i < n; i++){
        arr1[i] = rand()%100;
    }

    int k=999;
    clock_t begin = clock();
    #pragma omp parallel
    {
        #pragma omp for nowait
        for(i = 0; i < n; i++)
        {
            answer[i] = arr1[i] + j;
            printf("%d by thread %d\n",answer[i],omp_get_thread_num());
        }

        #pragma omp for nowait
        for(i = 0; i < n; i++)
        {
            answer2[i] = arr1[i] + k;
            printf("%d by thread %d\n",answer2[i],omp_get_thread_num());
        }

    }
    clock_t end = clock();
    double time_execution = (double)(end-begin)/CLOCKS_PER_SEC;
    printf("\nTime for execution: %lf", time_execution);
    return 0;
}
```



D:\academics\HPC Lab\ASSIGNMENT3\Q3_A_NoWait.exe

```
133 by thread 2
1033 by thread 2
99 by thread 3
999 by thread 3
168 by thread 4
1068 by thread 4
140 by thread 0
1040 by thread 0
166 by thread 1
1066 by thread 1

Time for execution: 0.002000
--------------------------------
Process exited after 0.6922 seconds with return value 0
Press any key to continue . . .
```

Github link: https://github.com/vai69/HPC-LAB